Chapter 1

Building Great iOS Games

In This Chapter

- Getting your mind on the game developer track
- Discovering the features that make for a good game
- Figuring out your game concept
- Fine-tuning the possibilities in your game design

Just as you find with any type of app, the range of games available for the iPhone (and iPad) is huge. They range from games that are expected to be chart busters from the beginning (the games produced by giant studios, such as Electronic Arts) to games made by individuals in their spare time that become huge hits (for example, *Trism* and *Flight Control*).

So, as a soon-to-be iPhone and iPad game creator, you need to find your slot in the range of games. As the authors of this book, we help you do that. In this chapter, we tell you how to get into the game developer mindset, determine what makes a good game, initiate a game concept, and then design the game to fully develop that concept.

When we started writing this book, we spent a lot of time figuring out the best way to showcase iOS game development. After much deliberation, we decided to showcase a complete game, dubbed *Traffic,* from start to finish. The alternative was to merely show you how to build pieces that could be useful in the development of a game. Instead, we chose to build a commercial-quality game step by step, demonstrating all the concepts and knowledge you need to build an amazing, real game of your own.

Enjoy!

COPYRIGHTED MATERIAL

iPhone, iPads, iDon'tKnows

The *iPad,* Apple's new computer, is fresh out of the factories and being bought by the millions. The iPad has been (somewhat unfairly) described as a "giant iPhone," which is inaccurate from a user experience point of view but rather accurate from a technical point of view.

Both the iPad and the iPhone run the same operating system — *iOS.* This means that 95 percent of the skills you pick up by reading this book apply to the iPad as much as they do to the iPhone. When we talk about iOS, or iOS devices, we're talking about the iPhone, the iPod touch, and the iPad.

Because these devices are so similar, whenever we refer to development on the iPhone, we also talk about development on the iPad as well.

Parts I, II, and III of this book discuss the development of the game for both the iPhone and the iPad. Part IV has more focus on the iPad and discusses the changes that you need to make so your game is the best it can be on the iPad.

Figuring Out What a User Wants from an iPhone Game

Think about a typical weekday — it's 8 a.m., and you're waiting for your train. You're bored. You've already checked your e-mail more times than is healthy, you've checked Twitter and told the world that your train is late, and you've checked the latest news headlines in your favorite news application. And you're still bored.

If only you had a game to pass the time! If you're using an iPhone, you probably do. You take your iPhone out of your pocket and touch the icon of your current favorite game to ease your boredom for a moment. Sixty seconds later, your bus arrives. You instantly snap out of the pocket-sized game world you were absorbed in, push your iPhone's Home button, and get on the bus.

On the train, you take a seat and pull your iPhone back out. Touching the icon of your favorite game again, you ease right back into play at exactly the same point you left off before you got on the train. Ten minutes later, your train pulls up at your stop, and you hit the Home button, pop the iPhone into your pocket, and head into work.

Why does all this matter? This scenario reflects the way most people play the best of the games available on the iPhone. They want to be able to listen to their music while they play, and they don't want the game to demand so much of them that they'll miss their train, or worse.

People play their iPhone games in potentially loud, bright, and distracting environments while they wait for something else to happen or while they talk to people. They play them for a minute or two before switching to something else, and they expect their iPhone to know what they were up to when they finally come back to the game.

Establishing a Game Developer Mindset

Why develop iPhone and iPad games? Because you can. Because it's time. And most of all, because it's fun! Developing a game that can potentially reach an audience in the millions is a hugely rewarding experience no matter how you look at it. Here's what makes developing games so much fun:

- **iOS games are usually small and conceptually simple to understand.** As with iPhone apps, a single developer, or maybe one with a partner and some graphics support, can do them. You don't need an enormous team with hordes of people, managers, and paperwork to create something rich and compelling. You have the power to create something that can reach millions, and you can do it from your own home.
- **Games on the iPhone and iPad are focused and clean.** The games get straight to the point of what makes them fun and help the users to dive in and out with ease. They're simple but not simplistic. This makes the design and implementation much easier and faster.
- **The popularity of the iOS platforms (that is, the iPhone and the iPad) makes getting your work into the hands of users easier than ever.** Getting your game onto a mobile device used to mean negotiating a deal with a publisher; these days, it's as simple as signing up online with Apple.

Before we talk about how to design your games, it's worth pointing out the single most valuable piece of advice one iOS game developer can give another: *Play other people's games!*

The more you play iOS games, the better you understand them. The better you understand them, the better your own games become. When you play, if you try to determine how the game actually works, you often strike inspiration. Many games appear simple on the surface, but if you delve deeper beneath the interface by paying closer attention to how you interact with the game and what the game presents to you in return, you reveal much hidden complexity in the way the game is constructed.

Discovering how others have built their games while you play them is the best way (other than reading this book) to develop your game building skills and gain a better understanding of what makes a great game tick.

Noting the Features of Good Games

Figure 1-1 shows the final version of the *Traffic* game you develop throughout this book. The concept for this game came to us after we noticed the popularity of simple puzzle games, line-drawing games, and solid (but simple), smartly presented game designs in the App Store.

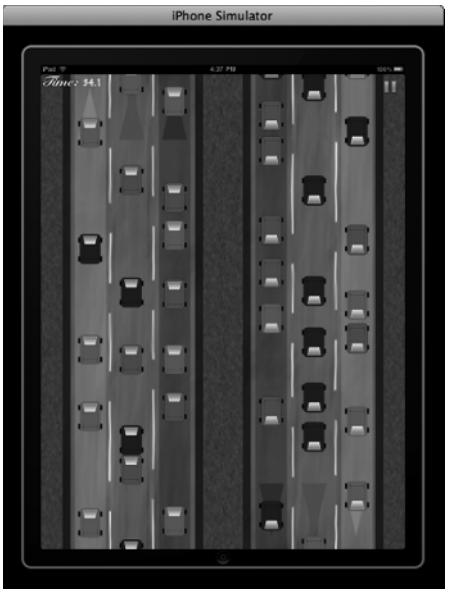

Figure 1-1: The *Traffic* game you build in this book.

How complex is the *Traffic* game? Not very. After you figure out how the game works in your head, and on paper, the actual programming doesn't take very long. Developing *Traffic* took us a little more than two months, working on and off.

Good iOS games share characteristics with good iOS applications of any kind. Before you jump in and design and build your game, make sure that you recognize these characteristics and incorporate them in your creation. We don't use *all* these characteristics in the *Traffic* game because it doesn't make any sense to simply cram ideas and features into a game in the spirit of embracing a platform. Judicious picking and choosing is essential to building a great game. In the next sections, we go over some of the most important.

Device-guided design

One of the keys to creating a great iOS application is to take advantage of the functionality that the device offers. In the case of a new platform (such as the iPhone 4 and the iPad), capitalizing on the new possibilities is especially important — especially when the application is a game!

Games are often expected to push the limits of a platform. When your game can easily incorporate new iOS (or hardware) functionality, new frontiers of game design and innovation open before you. These elements of iOS functionality — and how they relate to games — are as follows:

- **Accessing the Internet:** Allowing your games to offer users the ability to post their high scores to social networking sites, such as Facebook, or quickly and easily download new levels or content packs for your games is not just a good idea, it's essential. Word of your game spreads faster as users share their scores and favorite levels via their Facebook or Twitter pages. Your users also feel more connected and invested in your game because they're sharing it with their friends! By providing access to extra content stored online, your game's initial download size can also be made quite small.

  We cover making your game into a social beast in Chapter 16.

- **Detecting the location of the user:** Using the iPhone's built-in location services, you can determine the device's current location or even be notified when that location changes. In the context of gaming, location has a variety of potential uses — though many of them aren't obvious. For example, you could create a location-based game in which the player's location influences the game.

  Pac-Manhattan, a 2004 research project into location-aware games, had players running around the streets of New York carrying bulky GPS devices and re-creating a game of the arcade classic *Pac-Man.* Six years later, you have all the power of that hardware in your users' pockets.

- **Tracking motion and orientation:** The iPhone and iPad contain three accelerometers and a compass (and the iPhone 4 adds a gyroscope), which help you detect very small changes in movement. You can use these features to detect when the user turns the device from vertical to horizontal. In the case of iPhone games, you're probably more interested in subtle movements, such as tilting.

  Cro-Mag Rally, by Pangea Software, features a unique racing experience in which the user holds the iPhone like a steering wheel and turns it to drive the car. There are also a number of dexterity-based games in which the player must roll a ball around an obstacle course, such as *Super Monkey Ball* and *Labyrinth. Traffic* makes use of the accelerometer to detect the user shaking the device; you can read about how to add the feature to the game in Chapter 25.

- **Tracking multiple screen touches:** Because people use their fingers, rather than a mouse, to select and manipulate objects on the iPhone screen, take advantage of the fact that people have more than one finger! The iPhone can detect up to five individual fingers on the screen at any one time and lets you determine when people perform gestures with their fingers on the screen. The iPad can detect up to 11 individual touches on the screen simultaneously. (That's ten fingers plus your nose! We checked, using Jon's nose.)

  In games, gestures allow your players to have a very fluid and natural source of input to your game world. Flicking, pinching, and scrolling are very natural-feeling things to do in the iOS. If your game takes advantage of them, your users will notice, and they'll already know how to perform the most basic inputs to your game without needing a tutorial.

- **Playing audio and video:** The iOS makes playing and including audio and video in your application easy. You can play sound effects or take advantage of the multichannel audio and mixing capabilities available. You can also play back many standard movie file formats, configure the aspect ratio, and specify whether the controls are displayed.

  Of course, no game would be complete without a solid set of sound effects and a catchy theme tune! The iOS makes it easy to add these things as well as tweak the more complex and optional aspects of them, should the need arise.

- **Accessing the user's music library:** The iOS also makes gaining access to your user's songs, audio books, and audio podcasts very simple. You don't have to restrict your users to your game's theme music, but can allow them to pick and choose a custom playlist from their own library (or even assemble an entirely new playlist on the fly). This deceptively simple offering can help make your users feel more at home while playing your game and often entices them back to play more.

- **Accessing simple, ad hoc, location-based networking:** Specifically designed with games in mind, Apple's Game Kit allows you to create ad hoc Bluetooth networks among multiple iOS devices without the need for relatively complex Bluetooth pairing. This means your games can provide users with a very simple-to-activate multiplayer functionality, with the only requirement that they must be in proximity to another iPhone or iPad user running your game.

Incorporating the fun

Games need to be fun. When developing any game, examine several core principles of making the playing experience fun. There isn't a secret formula for games, but instilling and maximizing fun makes a better game for your users:

- **Happy players feel in control.** A lot of the fun in computer games is found in the pleasure of taking and manipulating the game world.
  - In *first-person shooter games* (combat-based games in which you have direct control over the way you move and the direction you look in), this manipulation takes the form of running around and shooting things. The player has control over what lives or dies in the game world but needs to be mindful of the dangers present in that world.
  - In *strategy games,* the player manipulates the world by sending units out to do battle, but also needs to be mindful of how and where to allocate these resources.

  In either case, a good game gives the players the feeling of control by reacting quickly to their input in a way that reflects what the player wants.
- **Happy players get surprised.** A game that's exactly the same every time has no replay value. A game in which you can anticipate enemy behavior after only a few seconds gets boring very fast. And so, another important component of a good game's fun factor is the amount that it surprises the player.

  An acceptable definition of fun itself could be *pleasure with surprises.*

  By combining the pleasure of being in control with an element of random chance, you can ensure that your game is neither too predictable nor too random.
- **Happy players find patterns.** As people play a good game, certain patterns of behavior emerge in the way they play. For example, in first-person shooters, the best players sidestep around corners, rather than turn around them, because sidestepping means that they can immediately aim and shoot at any threat around the corner. Clever game developers notice these patterns of play and find ways to improve the player's experience of them.

Designing a Good Game

Although jumping straight into code and getting down to building a game is exciting, clear and concise design is incredibly important in game development (perhaps even more so than it is to application development). Designing a game is a very rewarding experience. Although the frameworks and tools provided by Apple's iOS *Software Development Kit (SDK)* are vital to the process of building an iPhone or iPad game, knowing what you're going to build before you touch the SDK is just as vital.

Beginning with an idea

Game designs don't just spring into existence, fully formed. *Game design* is an organic process involving writing, reading, examining, rewriting, and updating. Go through the process of constructing an idea several times before you settle upon one.

A game concept starts to feel complete when it has the following:

- A description of the basic mechanics of the idea (how the game should play out and the basic actions that the player takes while playing)
- A basic story describing the motivation for the game play
- A flow (a basic game play description)
- Conceptual notes on graphics, feel, and audio
- Some examples of typical user interactions

So, how do you get these elements of your game concept in place? Well, the process somewhat depends on your game, and we can't really give you a blanket solution that works every time. But we can walk you through the steps of defining the elements as we did for the *Traffic* game. This process offers an understanding of the design decisions required and one method for arriving at them.

Making the idea fun, feasible, and unique

The idea for *Traffic* came from staring at the traffic passing and thinking, "That would make a fun game . . ." Of course, an iPhone game based on realistic simulation of traffic patterns wouldn't be too fun or accessible, so we had to pare down the idea to something that would work on the device. Line-drawing games have shown great longevity as popular titles at the App Store, so we approached the *Traffic* design with the idea that it'd be a line-drawing game.

In *line-drawing games,* the player sees an overall view of a scene and uses the iPhone's touchscreen to score points by drawing lines from one object to another (or a *goal* object). The genre has exploded in popularity, and you find many different variations on the general idea. Some great examples of line-drawing games that carry off the concept well are *Flight Control*, by Firemint, and *Harbor Master*, by Imangi Studios.

We didn't want *Traffic* to be just another line-drawing game, so we brainstormed further — striking upon the idea of a three-lane traffic system with different colored cars. As shown in Figure 1-2, the idea evolved over time, starting at a line-drawing game and ending at a traffic-swiping game. (We discuss how we evolved the game for the iPad in Chapter 19.)

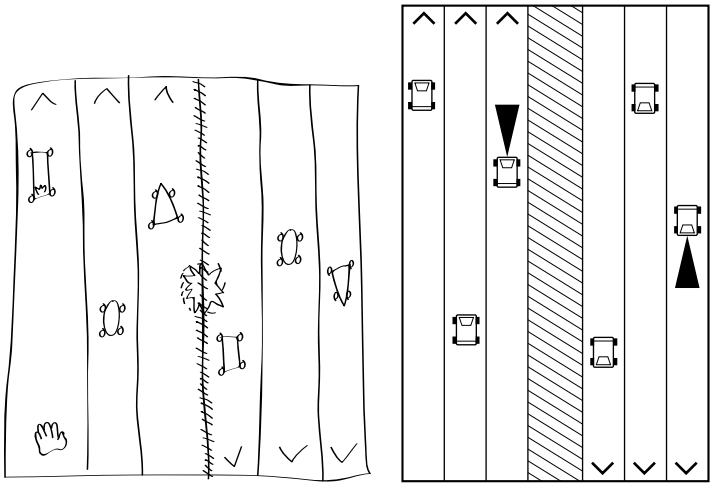

Figure 1-2: The evolution of the game from paper sketches.

Evolving the Game

No game idea comes fully formed, and it's important to try several approaches to a game concept before you commit your time to actual development. To do this, you must reduce the cost of throwing away ideas. And you'll throw away plenty of ideas. Trust us on that. One of the cheapest ways to try out ideas is to do so on paper.

Prototyping on paper

You may laugh, but drawing your game on paper (as shown in Figure 1-2) is one of the most important things you can do to make sure you're building a truly great game. So, how do you draw your game without feeling like a fool? And how do you make sure what you're drawing is useful?

To effectively prototype your game on paper, you need a few things; all are very cheap and easy to acquire. Here's the list.

- Lots of pencils, of various grades
- Some paper
- Some friends to "play" your paper game
- Patience, a good idea, and a sense of humor

After you collect what you need, sit down and think about the flow of your game. Think about even the most mundane things, such as menus and the game's launch. Here's the paper-prototyping process that you use to design the game flow, its mechanics, and its look and feel:

1. **Think about your game as a series of interconnected boxes of functionality and then draw those boxes and connections.**

   Start at the highest level you can go and distill the representation to the basic set of game functions you need to implement.

   As shown in Figure 1-3, making decisions about the flow of your game early is important.

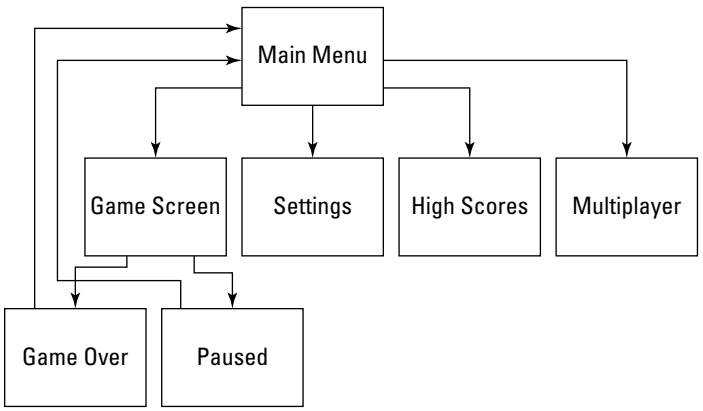

Figure 1-3: The flow of the *Traffic* screens.

2. **Draw the game board, and then add lines and arrows to show how objects move onscreen, as shown in Figure 1-4.**

   The simple act of drawing how game objects move and how they react to the user helps solidify how you see the game. These movements and reactions are the *game mechanics.* When designing your game's mechanics, consider how to keep the players busy without causing undue frustration.

   In general, giving the players the ability to do more means that the game maintains the player's interest. This is critical for games because if the player gets bored at any stage, the Home button is inches away from her fingers. Your game needs to be fun, intuitive, and exciting from the moment you launch the application.

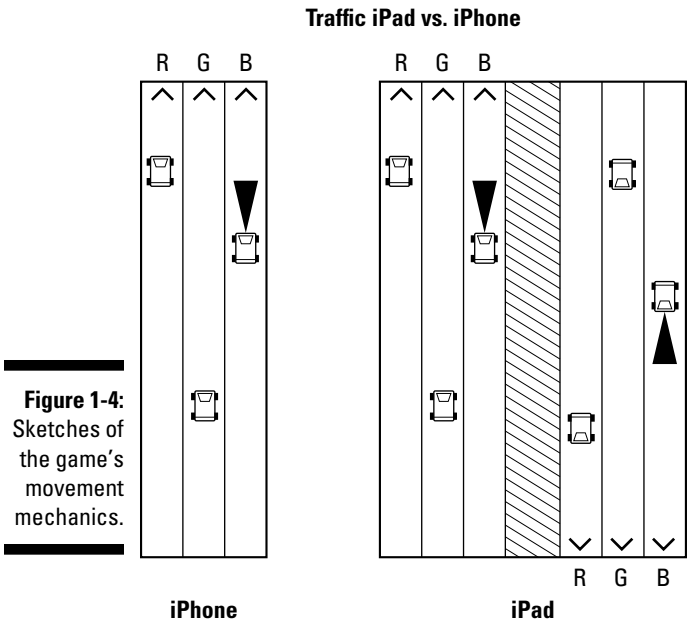

Figure 1-4: Sketches of the game's movement mechanics.

3. **Add the elements of style, color, and smaller graphical details that establish the look and feel for your game.**

   Figure 1-5 shows the finished visual prototype of the game's main menu.

   Ask yourself questions like the following, and create your look and feel accordingly. Do you want your game to look:

   - Simple or complex?
   - Realistic or cartoony?
   - Serious or funny?
   - Bright and cheerful, or dark and brooding?

Consider the amount of development time you have to invest (making the game art look realistic takes a large amount of time). Additionally, players expect things that look realistic to behave in realistic ways (which also takes time to code). In most games, the game designer is forced to make a trade-off over realism and fun, and we suggest that you err on the side of fun.

See the sidebar, "Moving through *Traffic*," nearby in this chapter, for some of the thought process we used to design the mechanics and look and feel for the *Traffic* game.

Moving through *Traffic*

The process of prototyping your game on paper gives you a great opportunity to think through how you want the game to work before you commit anything to code. Here are some of the thoughts we had while designing *Traffic:*

- Originally, we saw the game mechanics as being a choice among cars driving forward, cars turning, and the player directing traffic. However, this simply wasn't fun enough — more than half the cars didn't need to do anything to win points.
- There wasn't enough to indicate which cars should go where. We thought about adding blinking indicator lights on the cars but didn't feel that these would be visible enough.
- By adapting the game into three lanes going forward, we could have more cars onscreen at once without overloading the player in terms of the possibility of having them crash. This, in turn, allowed the player to concentrate on managing more cars.
- We decided on a simple, brightly colored theme and designed every aspect of the game's look around that. The buttons would be reminiscent of traffic lights, the cars would be seen from the top-down, and we'd keep the amount of clutter onscreen to a minimum.

Figure 1-5: The initial prototype of the main menu.

Distilling the ingredients of fun

There's no secret formula, ingredient, or blueprint for making your games fun. The hints and tips in this section help, but ultimately the only way to make a game fun is to tweak it until it's right.

Most players find games fun if they feel in control, can establish some patterns of play, and find that they're occasionally surprised by some element of the game. When you design a game, think through how to accomplish these characteristics of fun.

Giving a player control

When designing a game experience, figure out what the player controls. If an aspect of the game isn't controlled by the player, ask yourself whether the player could control it, even indirectly — and if he can't, would the game work without it?

Giving the player control can be a complex process and can take a lot of development time to fully implement. However, you can "cheat" in a couple ways and still have the player feel like she controls more than she actually does. For example, in the role-playing game *Mass Effect,* players can choose the flow of conversations by selecting the next line that they wish to say. However, having every conversation branch into every choice is simply too many options for the game developers to cover, but reducing the number of choices reduces the amount of control that they wanted the players to have.

The solution that the *Mass Effect* developers chose is quite elegant and simple: Instead of showing the exact line that the player's character would say, the game shows the intent of the next line. When the player chooses an intent, the line that their character would speak would be close to, but not quite, the text that the player chose.

The upshot of all this was that the developers could re-use lines of dialogue for different intents shown onscreen. By creating the illusion of choice, the player feels more involved; but in reality, the game developers didn't have to do any more work than they needed to.

Surprising a player

One simple way to add surprise is by adding random events to your game design. The venerable game *Missile Command* has a very simple rule set: Missiles fall from the sky, and the player must shoot them down. The fun comes from the random speed and direction that missiles fall. Players don't have infinite ammunition and can't afford to recklessly shoot everywhere, in hopes of getting every missile at once. The challenge (and surprise for the player) becomes anticipating where and how the missiles fall.

Encouraging patterns of play

Play your game (and have others play it) enough that you can pick out patterns of play. Then build responses to these patterns by adding a slight tilt to the screen, highlighting screen areas, adding subtle animations, and so on to make the player feel like his character is more involved in the game action. The effect is subtle but noticeable, and the game plays better for it.

A great example of one of these patterns is a side-scrolling game based on jump mechanics (think *Super Mario Brothers*, by Nintendo). After a certain amount of time playing, people become used to timing jumps as well as combining running and jumping. Observe this when you test your game with others; you can reward skilled jumping and running combinations, and work out new ways to test these skills.

If your game is so eclectic that your players can't find any patterns to improve their game with, take that as a signal that you need to add a little more structure to the game.

Applying Sid Meier's Rule of Halves

A lot of games rely on the finely tuned parameters, such as the speed of cars, the strengths of enemies, and the amount of ammo in your gun. These parameters often need to be just right — if they're not, the game feels wrong in difficult-to-define ways.

When trying to tune a game, the logical choice is to make small changes until it's right. Unfortunately, that's not possible when hundreds of factors are involved in a game — it'd simply take too much time. Thankfully, there's a solution.

Sid Meier, the legendary developer of such classics as *Civilization,* has a simple rule for tuning a game's parameters. If a parameter doesn't feel just right, either double it or cut it in half. If a car moves too fast, reduce its speed by half. If the gun feels too weak, double the amount of damage it does.

The point isn't that these new values are magically correct; in fact, you're likely to overshoot by a wide margin. The point is to narrow down the range of things to check. If your car is now too slow, change its speed to somewhere between its old speed and the speed it is now. Repeat this process until your parameters feel right.

Sid's rule of halves is quite a bit faster than the alternative, which can often involve plugging random numbers into your game code and seeing what works best; in fact, the math nerds among us will notice that it turns the time needed to figure out the best value from a linear equation to a logarithmic one. There's no arguing with science, kids.

Get your game out to other people: Show it to your friends, show it to strangers, and eventually, you need to bite the bullet and ship it. You won't find better feedback than from a paying customer's reaction; he won't pull his punches if he doesn't have fun with it.

What's Next

We're sure that you're raring to go now and just can't wait to download the SDK from the iPhone developer portal. That's exactly what each and every one of us did — when we first started development, we were ultra-keen and dived right into the code. Only later did we figure out that we needed to spend a little more time upfront understanding how games and applications work in the iOS environment.

We ask you to be patient! In Chapter 2, we explain what goes on behind the screen, and then, we promise, it's time to play in *Traffic*.

To make sure you're ready, head on over to the Web site at `http://traffic.secretlab.com.au` or `www.dummies.com/go/iphoneipadgameprogramming`. When you're there, click the big button that says "Download Resources". You'll get a zip file containing the imagery, audio, and other elements you'll need to build *Traffic*. Keep it safe and easily accessible, since we'll be referring to it a lot. The site also contains the latest version of each code listing, so if you get lost or just want to copy and paste the code instead of retyping it, make sure you grab that, too.